Le chiffrement à la volée

Anne Canteaut

INRIA - Projet CODES
BP 105
78153 Le Chesnay Cedex
http://www-rocq.inria.fr/~canteaut/

Les procédés de chiffrement à la volée sont des techniques qui permettent d'assurer la confidentialité d'une communication dans des contextes où il est primordial de pouvoir chiffrer et déchiffrer très rapidement et au moyen de ressources, notamment d'une capacité de stockage, très limitées. La plupart des systèmes embarqués entrent dans ce cadre, des communications téléphoniques aux communications satellites. Ces contraintes imposent donc l'usage d'algorithmes de chiffrement à clef secrète, puisque les algorithmes à clef publique connus actuellement n'offrent pas un débit suffisant pour permettre le chiffrement ou le déchiffrement en ligne.

Mais la structure dite "par blocs" des algorithmes à clef secrète utilisés dans de nombreuses applications, comme le DES ou l'AES, présente un inconvénient majeur pour certains systèmes embarqués. En effet, ces techniques consistent à découper le message à transmettre en blocs de taille fixe (généralement en blocs de 128 bits), puis à transformer par le même procédé chacun de ces blocs en un bloc de chiffré. De la même manière, le message reçu est déchiffré en appliquant successivement la transformation inverse à chacun de ses blocs. On ne peut donc commencer à chiffrer et à déchiffrer un message que si l'on connaît la totalité d'un bloc. Ceci occasionne naturellement un délai dans la transmission et nécessite également le stockage successif des blocs dans une mémoire-tampon. Au contraire, dans les procédés de chiffrement à la volée (appelés également techniques de chiffrement à flot, ou par flux), l'unité de base du chiffrement et du déchiffrement n'est plus le bloc mais le bit. Chaque nouveau bit transmis peut être chiffré ou déchiffré indépendamment des autres, en particulier sans qu'il soit nécessaire d'attendre les bits suivants. Un autre avantage de ces techniques est que, contrairement aux algorithmes par blocs, le processus de déchiffrement ne propage pas les erreurs de transmission. Supposons qu'une erreur survenue au cours de la communication ait affecté un bit du message chiffré. Dans le cas d'un chiffrement à la volée, cette erreur affecte uniquement le bit correspondant du texte clair, et ne le rend donc généralement pas complètement incompréhensible. Par contre, dans le cas d'un chiffrement par blocs, c'est tout le bloc contenant la position erronée qui devient incorrect après déchiffrement. Ainsi, une erreur sur un seul bit lors de la transmission affecte en réalité 128 bits du message clair. C'est pour cette raison que le chiffrement à la volée est également utilisé pour protéger la confidentialité dans les transmissions bruitées.

Le chiffre à usage unique

Les systèmes de chiffrement à la volée reposent sur le célèbre chiffre à usage unique, appelé aussi technique du masque jetable, et également connu sous son appellation anglo-saxonne "one-time-pad". Ce procédé, inventé par Vernam pour protéger les communications

télégraphiques pendant la première guerre mondiale, consiste simplement à effectuer un XOR (ou exclusif) bit à bit entre le message clair et une suite de bits aléatoire de même longueur qui constitue la clef secrète du système. Rappelons que l'opération XOR entre deux bits, représentée par le symbole $\oplus$, est définie par : $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$ et $1 \oplus 1 = 0$. Le chiffrement du message binaire 10100110 (représentation binaire de la lettre e) avec la clef secrète 01001011 se fait donc de la manière suivante :

message clair	1	0	1	0	0	1	1	0
	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$
clef secrète	0	1	0	0	1	0	1	1
message chiffré	1	1	1	0	1	1	0	1

Le déchiffrement est similaire : on retrouve le message d'origine à partir du message chiffré et de la clef par :

message chiffré	1	1	1	0	1	1	0	1
	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$
clef secrète	0	1	0	0	1	0	1	1
message clair	1	0	1	0	0	1	1	0

On peut démonter que le chiffre à usage unique est incassable dans la mesure où la connaissance du message chiffré n'apporte aucune information sur le message clair (cf. l'article de J. Patarin sur la cryptographie à clef secrète). Mais cette propriété n'est garantie que si la clef secrète est bien une suite totalement aléatoire aussi longue que le message clair, et qu'elle n'est utilisée que pour transmettre un seul message. L'emploi de la même clef pour chiffrer plusieurs messages peut notamment avoir des conséquences désastreuses (voir encadré sur le projet VENONA page 3). L'utilisation d'une clef secrète aléatoire à usage unique et de même longueur que le message à transmettre est malheureusement nécessaire pour obtenir un chiffrement inconditionnellement sûr, c'est-à-dire pour lequel on peut prouver qu'il est impossible de retrouver le message clair à partir du chiffré sans connaître la clef secrète. Cette condition rend généralement tous les chiffrements parfaits, comme le chiffre à usage unique, inutilisables puisqu'il est rarement envisageable de s'échanger préalablement un secret aussi long que la totalité des messages à transmettre. L'usage de ces systèmes est donc réservé aux communications exigeant un niveau de sécurité extrêmement élevé comme les communications diplomatiques (le canal sécurisé utilisé pour transmettre les suites aléatoires secrètes n'est autre que la valise diplomatique). C'était le cas par exemple du célèbre "téléphone rouge" entre Washington et Moscou pendant la guerre froide.

Les générateurs pseudo-aléatoires cryptographiques

Dans les applications usuelles, le chiffrement à la volée utilisé est une version affaiblie du chiffre à usage unique : la suite que l'on additionne par XOR au message clair n'est plus une suite engendrée de manière totalement aléatoire, mais une suite *pseudo-aléatoire*. Cette suite est engendrée de manière déterministe par un dispositif appelé générateur pseudo-aléatoire qui est initialisé par une valeur secrète relativement courte (de l'ordre de 128 bits) et qui correspond à la clef du système. Ce type de procédé ne garantit donc plus une sécurité théorique : la connaissance du chiffré apportera toujours théoriquement une information sur le message clair. Par contre, cette information peut être très difficile à obtenir (décrypter le message

Le projet VENONA

L'exemple le plus célèbre d'une mauvaise utilisation du chiffre à usage unique est celle qu'en a faite le KGB soviétique pendant la seconde guerre mondiale. Les transmissions soviétiques étaient alors chiffrées de la façon suivante : un dictionnaire permettait de coder les lettres ainsi que certains mots ou certaines phrases par des nombres, qui étaient ensuite chiffrés par un chiffre à usage unique. Mais certains opérateurs, notamment en 1944 et 1945, ont eu l'imprudence d'utiliser régulièrement la même suite secrète pour chiffrer plusieurs messages. Cette erreur grossière a notamment permis à la National Security Agency (NSA) américaine de décrypter des télégrammes soviétiques de première importance. Un projet secret désigné par le nom de code VENONA a été mis en place en 1943 par la NSA et le FBI, rejoints par les services anglais en 1948 ; il était chargé de décrypter les communications échangées entre Moscou et certaines ambassades soviétiques à partir de 1939. Son existence et le contenu de divers messages décryptés dans ce cadre n'ont été révélé par la NSA qu'à la fin des années 90 (certains d'entre eux sont reproduits sur le site Web de la NSA `http://www.nsa.gov/docs/venona/` et sur celui de la CIA `http://www.cia.gov/csi/books/venona/venona.htm`). Parmi ces messages, on trouve notamment des informations concernant les réseaux d'espionnage à caractère technologique, notamment l'espionnage nucléaire. Ainsi, un télégramme transmis le 27 novembre 1944 de New-York à Moscou, apporte des renseignements sur l'espion appelé LIBERAL : *Informations sur l'épouse de LIBERAL : prénom Ethel, 29 ans, mariée il y a 5 ans, membre du parti communiste depuis 1938. [...] Au courant des travaux de son mari.* Ce télégramme, et d'autres, ont été considérés comme des preuves à la charge des époux Rosenberg.

peut nécessiter par exemple plusieurs milliers d'années sur un grand nombre d'ordinateurs en parallèle). On se place donc ici sur le plan de la sécurité calculatoire puisqu'on évalue la sécurité du système par le temps de calcul et la quantité de ressources nécessaires pour le cryptanalyser.

La sécurité d'un algorithme de chiffrement à la volée repose entièrement sur les qualités cryptographiques du générateur pseudo-aléatoire employé. Celui-ci doit notamment résister à une attaque à clair connu, dans laquelle un attaquant dispose de divers messages chiffrés ainsi que des textes clairs correspondant — qu'il a devinés par exemple en utilisant le fait que les messages envoyés suivent un format particulier. Un simple XOR entre ces couples clairs-chiffrés fournit alors les valeurs de certains bits produits par le générateur. Dans ce contexte, il doit donc être impossible en pratique quand on connaît certains bits produits par le générateur de prédire la valeur des bits suivants. Autrement dit, cette propriété est celle que l'on exigerait si un tel procédé était utilisé pour tirer les numéros gagnants au Loto : une personne qui analyse tous les tirages du Loto doit être dans l'impossibilité matérielle d'en déduire une quelconque information sur les tirages à venir.

Les générateurs à base de registres à décalage

Une idée relativement naturelle de générateur pseudo-aléatoire qui soit à la fois très rapide et solide est de générer une suite par une relation de récurrence linéaire. La clef secrète correspond aux k premiers bits de la suite $s_1, s_2, \ldots, s_k$, et chacun des bits suivants est ensuite calculé par récurrence en effectuant un XOR entre certains des k bits précédents : $s_t = c_1 s_{t-1} \oplus c_2 s_{t-2} \oplus \ldots \oplus c_k s_{t-k}$ où les coefficients c_i sont fixés et valent 0 ou 1. Le grand avantage de cette méthode est qu'une telle suite peut être engendrée au moyen d'un dispositif électronique très simple, très rapide et peu coûteux appelé registre à décalage à rétroaction linéaire. Il s'agit d'un registre contenant k bits — au départ il sera initialisé par les valeurs $s_1, s_2, \ldots, s_k$. A chaque instant $t > k$, on calcule à partir du contenu du registre un nouveau bit, appelé bit de rétroaction, par la formule $c_1 s_{t-1} \oplus c_2 s_{t-2} \oplus \ldots \oplus c_k s_{t-k}$. Puis, le bit de droite sort du registre et les autres sont décalés vers la droite. Le bit de rétroaction que l'on vient de calculer est alors placé dans la case la plus à gauche. La figure 1 décrit l'évolution d'un registre de longueur 5 correspondant à la récurrence $s_t = s_{t-2} \oplus s_{t-5}$ et initialisé par les valeurs $s_1 = 0$, $s_2 = 1$, $s_3 = 0$, $s_4 = 0$, $s_5 = 1$.

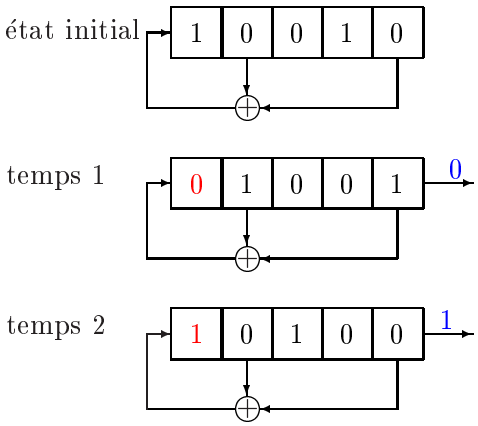
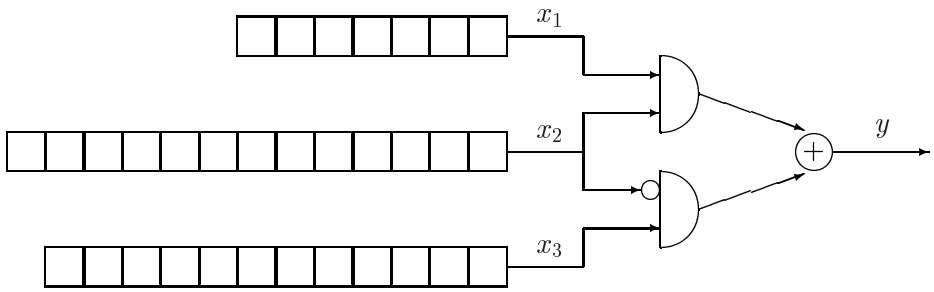


FIG. 1 – *Exemple de fonctionnement d'un registre à décalage à rétroaction linéaire*

On ne peut évidemment pas utiliser ce dispositif tel quel comme générateur pseudo-aléatoire puisque la connaissance de k bits consécutifs de la suite produite permet de déterminer tous les suivants. Mais il est possible de combiner plusieurs registres de ce type pour engendrer une suite ayant de bonnes propriétés cryptographiques. Par exemple, le générateur proposé par P. Geffe en 1973 produit une suite calculée à partir de trois registres différents : on effectue un ET logique entre les bits x_1 et x_2 produits respectivement par le premier et le deuxième registres, et un ET logique entre le bit x_3 produit par le troisième registre et le complémentaire de x_2. Le bit produit par le générateur est donné par un XOR entre les résultats des deux ET précédents. Autrement dit, le bit de sortie est égal à $y = (x_1 \text{ AND } x_2) \oplus (\overline{x_2} \text{ AND } x_3)$. La clef secrète est constituée par l'ensemble des initialisations des trois registres. Sa taille correspond donc à la somme des longueurs des registres. Cependant, même lorsque cette taille met le système à l'abri d'une recherche exhaustive (par exemple, si la clef fait 128 bits), il est très facile de cryptanalyser ce générateur comme l'a montré T. Siegenthaler en 1985. En effet, on peut exploiter le fait que sur les 8 triplets (x_1, x_2, x_3) possibles, 6 d'entre eux sont tels que la sortie y du générateur coïncide avec la

valeur de x_1. Cette corrélation entre la sortie du générateur de Geffe et la suite engendrée par le premier registre permet de mener une attaque de type "diviser pour mieux régner", qui consiste à retrouver séparément (et non plus simultanément) les initialisations des trois registres (voir l'encadré sur l'attaque par corrélation du générateur de Geffe). Il existe des variantes plus récentes et plus performantes de ce type d'attaques. Elles sont extrêmement dévastatrices, ont un vaste champ d'application et imposent des contraintes très strictes dans la conception de ce style de générateurs pseudo-aléatoires, notamment dans le choix de la fonction choisie pour combiner les différents registres.

Attaque par corrélation du générateur de Geffe



Considérons par exemple un générateur de Geffe composé de trois registres de longueurs respectives 7, 12 et 13. Une recherche exhaustive sur la clef secrète, qui comporte $7 + 12 + 13 = 32$ bits, nécessiterait d'examiner ses $2^{32} = 4.294.967.296$ valeurs possibles. En fait, on peut déterminer de manière indépendante les 7 premiers bits de la clef, qui correspondent à l'initialisation du premier registre. On considère les $2^7 = 128$ initialisations possibles pour ce registre. Pour chacune d'entre elles, on génère les 100 premiers bits produits par le premier registre initialisé de la sorte et on compare ces 100 bits avec les 100 premiers bits produits par le générateur. Si l'initialisation du premier registre que l'on essaie est correcte, ces deux suites de 100 bits vont coïncider sur environ 75 bits puisque la sortie du générateur est égale à la sortie du premier registre dans 75 % des cas. Par contre, si l'initialisation essayée n'est pas la bonne, les deux suites que l'on compare ne sont pas corrélées. Autrement dit, elles vont coïncider sur environ la moitié des positions. Ainsi, la connaissance des 100 premiers bits produits par le générateur suffit à déterminer l'initialisation du premier registre en 128 essais. De la même façon, on peut ensuite retrouver l'initialisation du troisième registre (de longueur 12) en $2^{12} = 4096$ essais en exploitant le fait que la sortie du générateur coïncide également dans 75 % des cas avec la suite engendrée par le troisième registre. Enfin, il ne reste plus qu'à retrouver les 13 bits de clef restant à déterminer (l'initialisation du deuxième registre) par une recherche exhaustive. L'attaque complète du générateur a donc nécessité $2^7 + 2^{12} + 2^{13} = 12.416$ essais au lieu des $2^{32} = 4.294.967.296$ demandés par une recherche exhaustive de la clef.

L'algorithme de chiffrement du GSM

Les générateurs pseudo-aléatoires reposant sur des registres à décalage à rétroaction linéaire qui sont utilisés dans la pratique sont généralement plus complexes que le générateur de Geffe. C'est par exemple le cas de l'algorithme A5/1 utilisé en Europe pour chiffrer les communications GSM — une variante beaucoup plus faible de ce système, appelée A5/2, est utilisée d'autres pays. Les spécifications de l'algorithme A5/1 ne sont pas publiques, mais elles ont été retrouvées par un processus de reverse engineering et divulguées sur Internet. Dans une communication GSM, la voix est numérisée, compressée puis transmise sous forme de trames de 114 bits. Environ 217 trames sont échangées entre le téléphone mobile et le réseau lors d'une seconde de communication GSM. Chaque couple de trames (une du mobile vers le réseau et une du réseau vers le mobile) est chiffré par un XOR bit à bit avec une suite de 228 bits produite par un générateur pseudo-aléatoire initialisé par une clef secrète de 64 bits (qui a été établie quand l'utilisateur s'est connecté au réseau GSM) et un numéro de trame (qui est un nombre public de 22 bits). Le générateur utilisé dans A5/1 est composé de trois registres à décalage à rétroaction linéaire de longueurs respectives 19, 22 et 23. Pour chaque couple de trames, on initialise ce générateur de la façon suivante : chacun des registres est initialisé à zéro à l'instant $t = 0$. Puis, aux instants $t = 1 \dots 64$, on remet à jour en parallèle les 3 registres en ajoutant par XOR au bit de rétroaction le bit t de la clef secrète. On répète ce processus aux instants $t = 65 \dots 86$ en ajoutant maintenant aux bits de rétroaction le bit $(t - 64)$ du numéro de trame. Le générateur est initialisé à la fin de ces 86 cycles et peut entrer dans son fonctionnement normal.

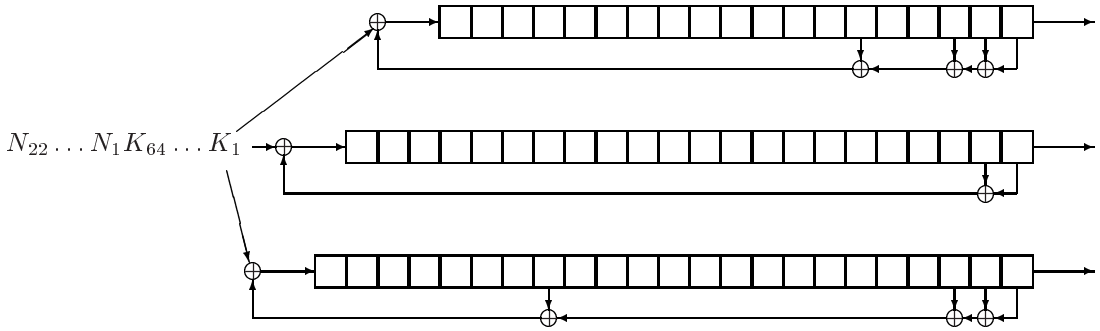


FIG. 2 – *Procédure d'initialisation du générateur utilisé par A5/1*

Celui-ci est caractérisé par une procédure particulière de contrôle de l'horloge des registres. Chacun des registres possède un bit de contrôle d'horloge : le bit 11 pour le premier registre, le bit 12 pour le second et le bit 13 pour le troisième. A chacun instant, on calcule la valeur b qui apparaît majoritairement parmi ces trois bits et, pour chacun des registres, on compare la valeur de son bit de contrôle d'horloge et celle de b. Si ces deux valeurs sont identiques, le registre correspondant sera remis à jour à l'instant suivant ; sinon, son état restera inchangé. Ainsi, dans l'exemple de la figure 3, seuls les deux premiers registres sont modifiés. La sortie du générateur correspond alors au XOR des sorties des trois registres remis à jour de cette manière. On génère par ce procédé 328 bits. Les 100 premiers bits ne sont pas utilisés ; les 114 suivants sont ajoutés par XOR à la trame transmise par le mobile et les 114 derniers à la trame transmise par le réseau.

La solidité de l'algorithme A5/1 s'avère suffisante pour dissuader certains pirates mais

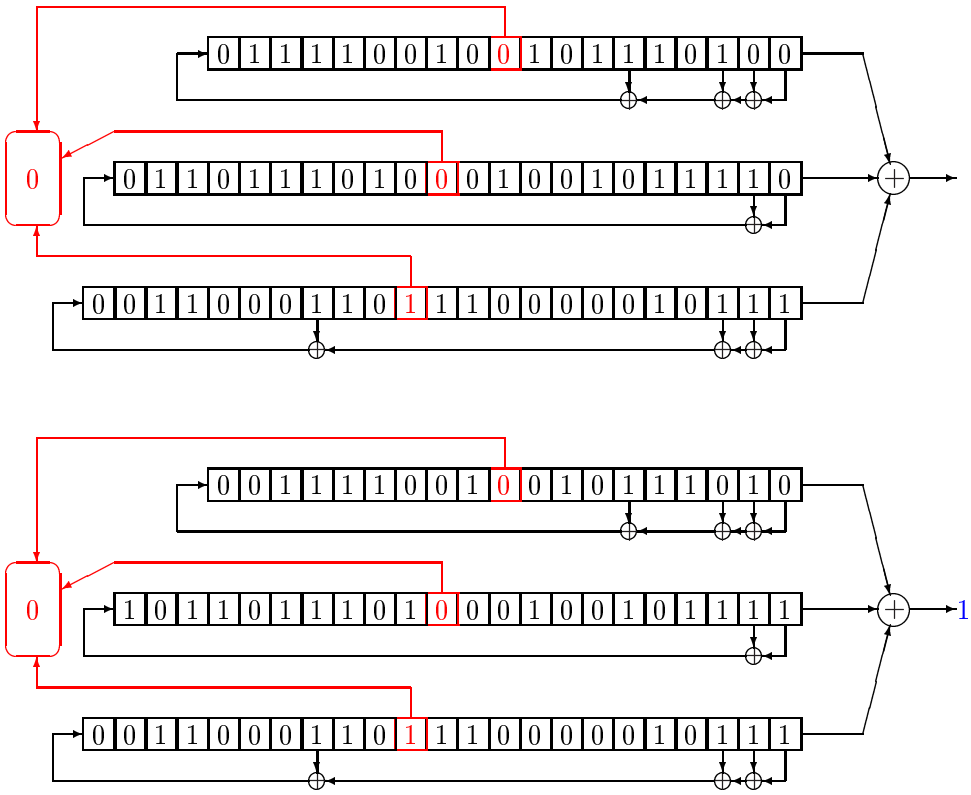


FIG. 3 – *Fonctionnement du générateur utilisé par A5/1*

elle ne permet pas de se mettre à l'abri d'un attaquant possédant une puissance de calcul importante. En effet, A. Biryukov, A. Shamir et D. Wagner ont montré qu'il est possible de retrouver la clef secrète utilisée pour initialiser le générateur dès lors que l'on connaît suffisamment de couples de trames claires-chiffrées. Cette cryptanalyse nécessite au préalable un calcul relativement coûteux en temps et en mémoire, mais qui se fait une fois pour toutes. Après cette phase de précalcul, il est possible de retrouver la clef secrète en une seconde de calcul sur un PC si l'on a accès à deux minutes de communication. Deux secondes de communication permettent également de retrouver la clef, mais cette fois-ci après plusieurs minutes de calcul.

L'algorithme de chiffrement de l'UMTS

La relative faiblesse de l'algorithme A5/1 a amené à modifier complètement l'algorithme de chiffrement pour les téléphones mobiles de troisième génération (UMTS). Au contraire des générateurs pseudo-aléatoires décrits précédemment, le système de chiffrement de l'UMTS, F8, n'est pas construit à partir de registres à décalage mais à partir d'un chiffrement par blocs. En effet, tout algorithme par blocs permet d'engendrer une suite pseudo-aléatoire. Cette utilisation particulière, appelée mode opératoire OFB (Output FeedBack), consiste à chiffrer une valeur initiale au moyen de l'algorithme par blocs paramétré par la clef secrète du système. Le chiffré correspondant fournit le premier bloc de la suite pseudo-aléatoire. Ensuite,

chaque nouveau bloc de suite est obtenu en chiffrant par le même procédé le résultat du XOR entre le bloc de suite précédent et la valeur initiale.

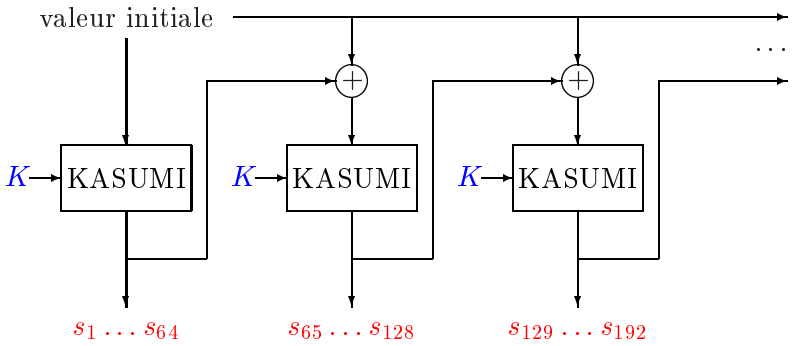


FIG. 4 – *Le générateur pseudo-aléatoire de F8*

L'inconvénient majeur de ce type de procédé est qu'il est plus lent que les générateurs à base de registres. Mais sa sécurité repose essentiellement sur la solidité de l'algorithme par blocs utilisé. Celui employé dans F8 est l'algorithme KASUMI, qui est une variante de l'algorithme MISTY1 conçu par M. Matsui. Il s'agit d'un chiffrement de Feistel opérant sur des blocs de 64 bits avec une clef secrète de 128 bits, et pour lequel on ne dispose actuellement d'aucune attaque significative.

Références

[1] A.J. MENEZES, P.C. VAN OORSCHOT, et S.A. VANSTONE. « *Handbook of Applied Cryptography* », Chapitre 6 : Stream ciphers. CRC Press, 1996. Disponible sur `http://cacr.math.uwaterloo.ca/hac/`.

[2] L. TARKKALA. « Attacks against A5 ». Dans *Proceedings of the Helsinki University of Technology seminar on network security: Mobile Security*, 2000. Disponible sur `http://www.tcm.hut.fi/Opinnot/Tik-110.501/2000/papers/`.